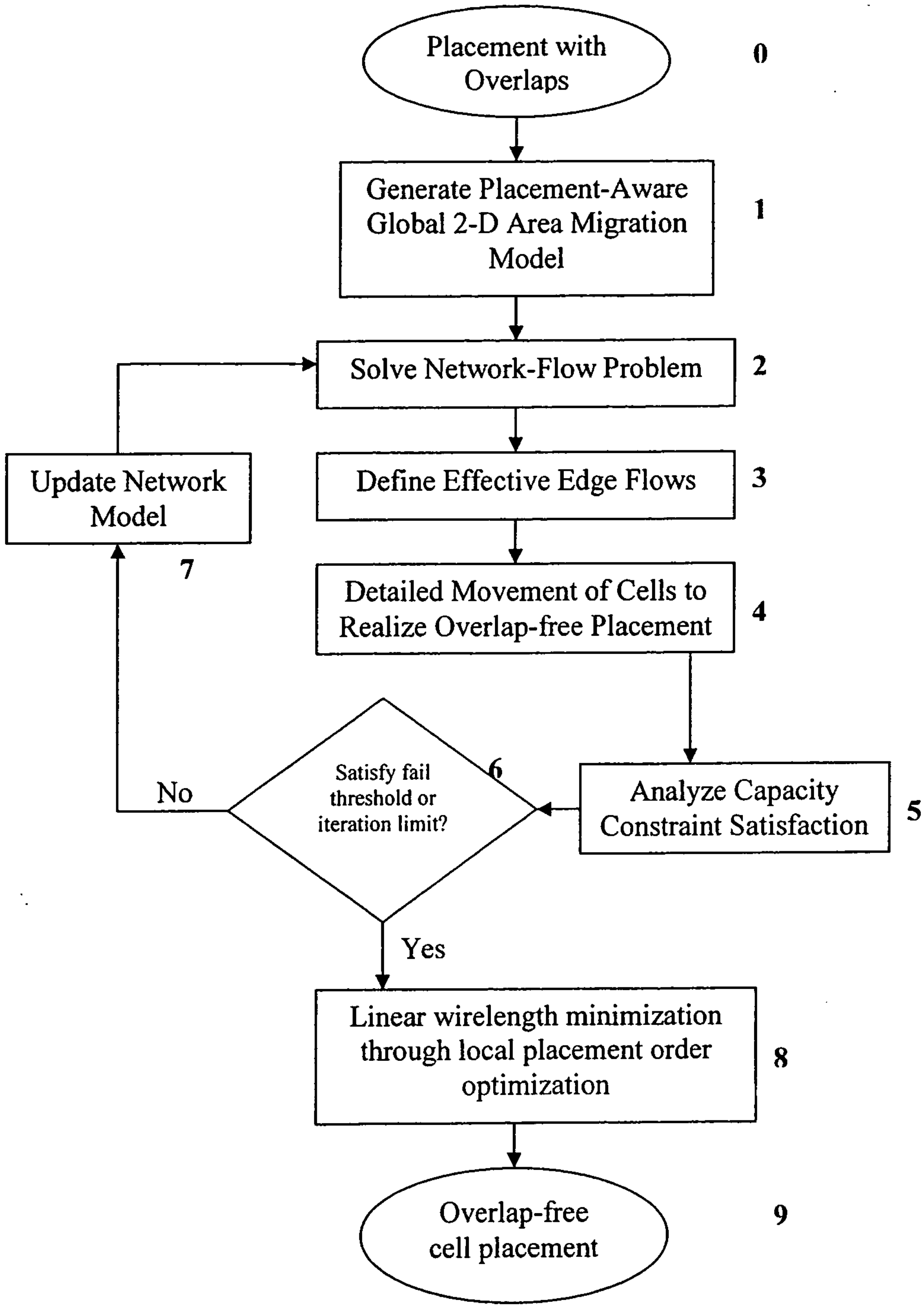

Placement with Overlaps
0
Generate Placement-Aware Global 2-D Area Migration Model
1
Solve Network-Flow Problem
2
Update Network Model
Define Effective Edge Flows
3
7
Detailed Movement of Cells to Realize Overlap-free Placement
4
Satisfy fail threshold or iteration limit?
6
No
Analyze Capacity Constraint Satisfaction
5
Yes
Linear wirelength minimization through local placement order optimization
8
Overlap-free cell placement
9

Figure 1

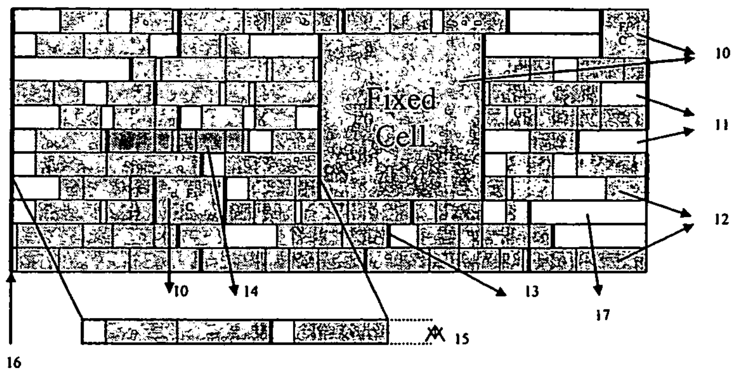


Figure 2

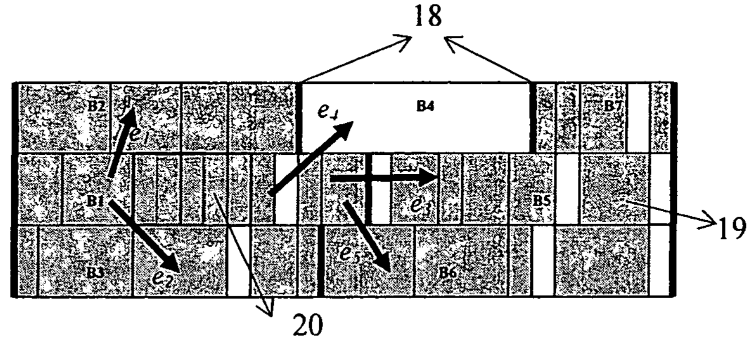
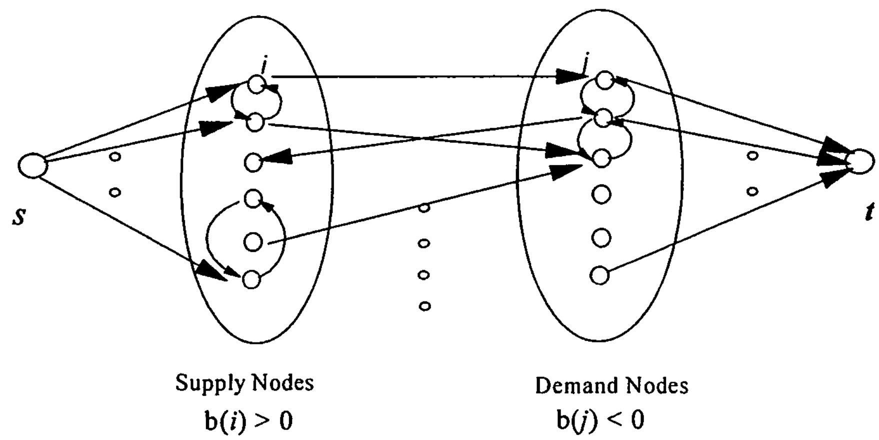


Figure 3



$\forall\, i$ *if* b(i) > 0,
Cap(e_{si}) = b(i)
Cost(e_{si}) = 0

$\forall\, i \neq s$, $j \neq t$,
Cap(e_{ij}) = *Infinity* (Large Int)
Cost(e_{ij}) = $K_{e_{ij}}$

$\forall\, j$ *if* b(j) < 0,
Cap(e_{jt}) = -b(j)
Cost(e_{jt}) = 0

$\forall$: Notation represents the meaning *"For Every Element"*
$\in$: Notation represents the meaning *"Element of"*

Figure 4

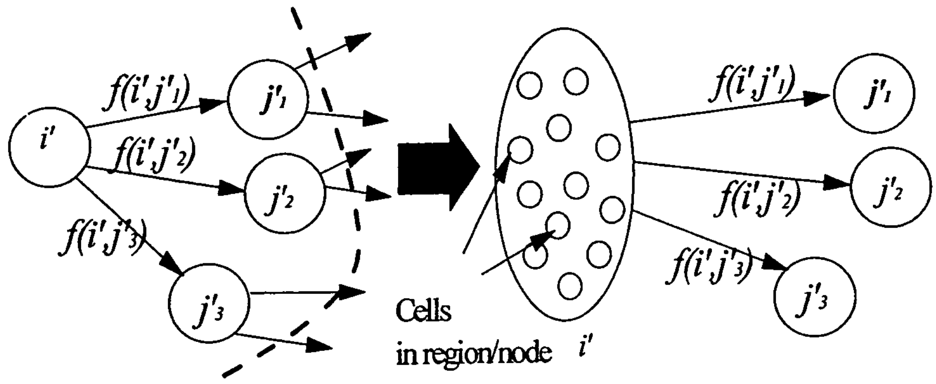
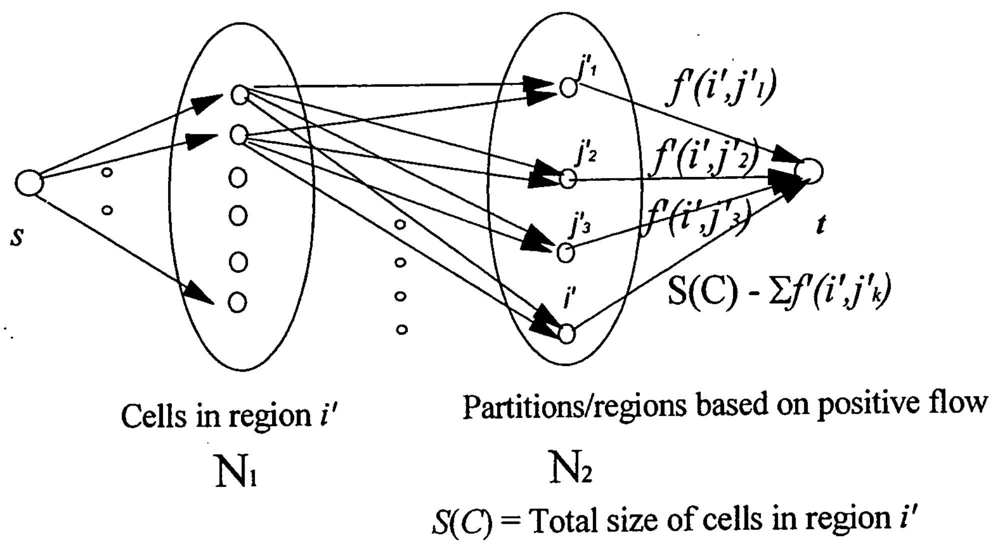


Figure 5



$\forall i \in N_1$
Cap(e_{si}) = 1
Cost(e_{si}) = 0

$\forall i \in N_1, j \in N_2,$
Cap(e_{ij}) = *1*
Cost(e_{ij}) = *Cost of moving cell i to region j*
multiplier μ_{ij} = *size of cell i*

$\forall j \in N_2$
Cap(e_{jt}) = *flow to region j*
Cost(e_{jt}) = 0

$\forall$: Notation represents the meaning *"For Every Element"*
$\in$: Notation represents the meaning *"Element of"*

Figure 6

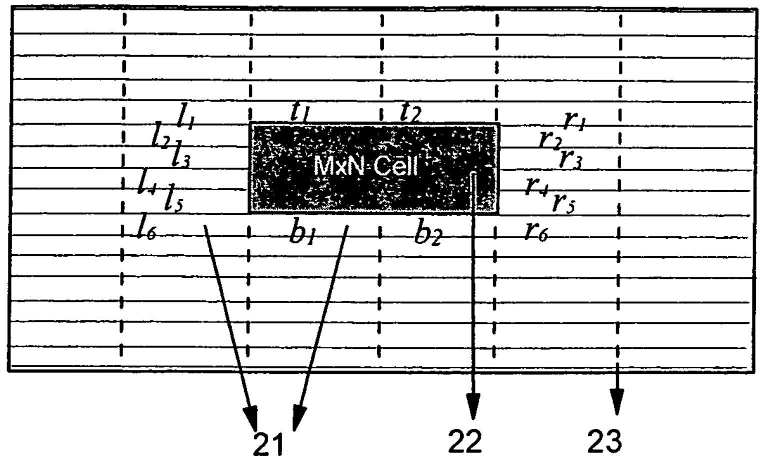
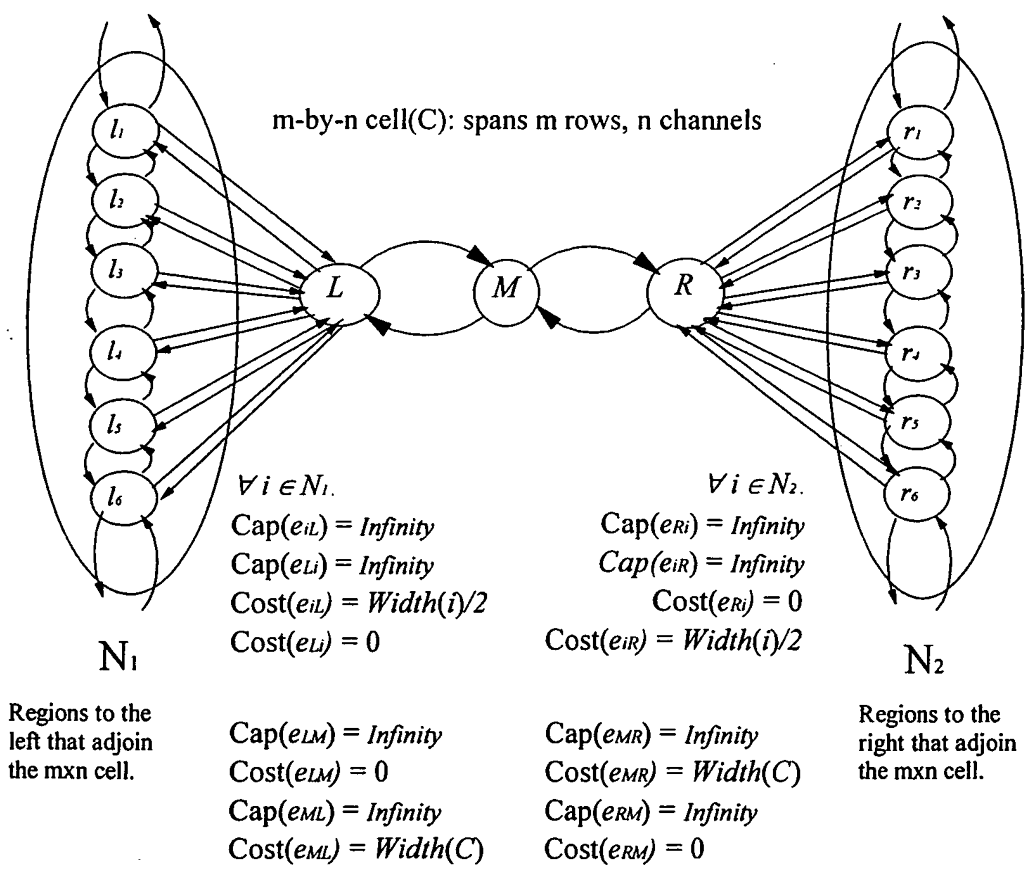


Figure 7



$\forall$: Notation represents the meaning *"For Every Element"*

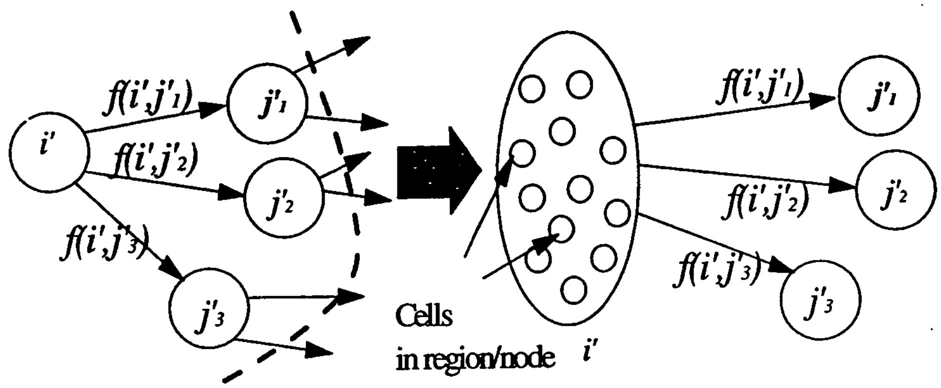
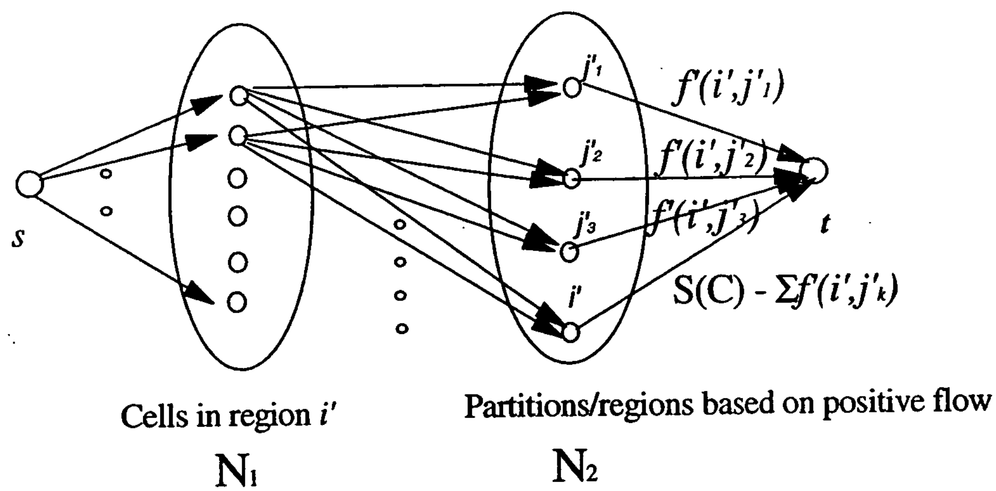


Figure 5



$S(C)$ = Total size of cells in region i'

$\forall i \in N_1$
$\mathrm{Cap}(e_{si}) = 1$
$\mathrm{Cost}(e_{si}) = 0$

$\forall i \in N_1, j \in N_2,$
$\mathrm{Cap}(e_{ij}) = 1$
$\mathrm{Cost}(e_{ij})$ = *Cost of moving cell i to region j*
multiplier μ_{ij} = *size of cell i*

$\forall j \in N_2$
$\mathrm{Cap}(e_{jt})$ = *flow to region j*
$\mathrm{Cost}(e_{jt}) = 0$

$\forall$: Notation represents the meaning *"For Every Element"*
$\in$: Notation represents the meaning *"Element of"*

Figure 6

```
Placement_Aware_Region_Definition ()
Begin
 1 Build placement image
 2 For each circuit row r in the layout
   Begin
 3      scanline_x = row_xlow (r); last_region_boundary = row_xlow (r);
 4      leading_free_space = false;
 5      S = sorted list of cells in row r by increasing position along x-direction
        c = first cell in sorted list S
 6      while (c)
          Begin
 7          If (xpos(c) > scanline_x)
            Begin
 8            If (xpos(c) - last_region_boundary > W)
              Begin
 9              p = create_region (r,last_region_boundary+W, last_region_boundary)
10              scanline_x = (last_region_boundary+ = W)
              End
11            Else
              Begin
12              If (is_fixed_cell(c) || is_blockage(c) || leading_free_space)
                Begin
13                p = create_region (r, xpos(c), last_region_boundary)
14                scanline_x = last_region_boundary = xpos(c)
15                leading_free_space = false
                End
16              Else if (xpos(c) - scanline_x >= 0.50 * W and scanline_x > last_region_boundary)
                Begin
17                p = create_region (r, scanline_x, last_region_boundary)
18                last_region_boundary = scanline_x
19                leading_free_space = true
                End
20              Else   scanline_x = xpos(c)
              End
            End
21.         Else if (xpos(c) == scanline_x)
            Begin
22            If (is_fixed_cell(c) || is_blockage(c))
              Begin
23              p = create_region (r, xpos(c) + width(c), scanline_x)
24              scanline_x += width(c)
25              last_region_boundary = scanline_x
              End
26            Else if (is_movable_cell(c))
              Begin
27              If (xpos(c) + width(c) <= W)
28                scanline_x += width(c)
29              Else
                Begin
30                p = create_region (r, xpos(c) + width(c), last_region_boundary)
31                last_region_boundary = scanline_x
32                scanline_x += width(c)
                End
              End
33          c= next cell in the sorted list S
            End
          End
   End
End
```

FIGURE 9

Global_Area_Migration_Graph ($G(V,E)$)

Begin

1. V = {regions}, E = {edge between neighboring regions}
2. $\forall\, e \in E$, Cost(e) = K_e
3. $\forall\, e \in E$, Cap(e) = *Infinity* (Large integer)
4. $\forall\, v \in V$, Size(v) = Total size of movable cells in v
5. $\forall\, v \in V$, Cap(v) = Total available space for movable cells in v (i.e. region)
6. $\forall\, v \in V$, b(v) = Size(v) - Cap(v)
7. If b(v) > 0, v is a supply node.
8. If b(v) < 0, v is a demand node.
9. If b(v) = 0, v is a transshipment node.

End

$\forall$: **Notation represents the meaning *"For Every Element"***

$\in$: **Notation represents the meaning *"Element of"***

Figure 10

Generalized_Flow_Graph (region *i'*)

Begin

1. N_1 = {cells in region i'}, N_2 = {i'} U {*neighboring regions*}
2. E = {edge representing cell-to-region assignment}
3. S(N_1) = Total size of cells in N_1 (region i')
4. *Smallest*(N_1) = Smallest cell size in N_1 (region i')
5. *Introduce an edge from N_1 to N_2 for every possible cell-to-region assignment,*

   $\forall i \in N_1, j \in N_2$, Cap($e_{ij}$) = 1

   $\forall i \in N_1, j \in N_2$, *multiplier*, μ_{ij} = *size of cell i*

   $\forall i \in N_1, j \in N_2$, Cost($e_{ij}$) = *Cost of moving cell i to region j*
6. *Introduce source node s, with edges such that*

   $\forall i \in N_1$, Cap(e_{si}) = 1

   $\forall i \in N_1$, Cost(e_{si}) = 0
7. *Introduce sink node t, with edges such that*

   $\forall j \in N_2$, Cap(e_{jt}) = $f'(i',j)$ = *MAX(Smallest*(N_1), $f(i',j)$), *If* $f(i',j) > 0$

   0, Otherwise

   $\forall j \in N_2$, Cost(e_{jt}) = 0

End

$\forall$ **: Notation represents the meaning** ***"For every element"***

$\in$ **: Notation represents the meaning** ***"Element of" (a set theory notation)***

Figure 11